Time series forecasting by evolving artificial neural networks using genetic algorithms and differential evolution

Juan Peralta, Xiaodong Li, German Gutierrez, Araceli Sanchis

***Abstract*—Accurate time series forecasting are important for displaying the manner in which the past continues to affect the future and for planning our day to-day activities. In recent years, a large literature has evolved on the use of evolving artificial neural networks (EANNs) in many forecasting applications. Evolving neural networks are particularly appealing because of their ability to model an unspecified non-linear relationship between time series variables. This paper evaluates two methods to evolve neural networks architectures, one carried out with genetic algorithm and a second one carry out with differential evolution algorithm. A comparative study between these two methods, with a set of referenced time series will be shown. The object of this study is to try to improve the final forecasting getting an accurate system.**

I. Introduction

IN order to acquire knowledge, it is interesting to know what the future will look like, i.e. forecast the future from the observed past. Time series forecasting is an essential research field due to its effectiveness in human life. It is a discipline that finds each day more applications in areas like planning, management, production, maintenance and control of industrial processes, economy, and weather forecasting.

The forecasting task can be performed by several techniques as Statistical methods [1], and others based on Computational Intelligence like Immune Systems [2] and Artificial Neural Networks (ANN) [3].

ANNs provide a methodology for solving many types of nonlinear problems that are difficult to solve by traditional techniques. Most time series processes often exhibit temporal and spatial variability, and are suffered by issues of nonlinearity of physical processes, conflicting spatial and temporal scale and uncertainty in parameter estimates. The ANNs have capability to extract the relationship between the inputs and outputs of a process, without the physics being explicitly provided. Thus, these properties of ANNs are well suited to the problem of time series forecasting.

The research reported here has been supported by the Spanish Ministry of Science and Innovation under project TRA2007-67374-C02-02.

J. Peralta. Computer Science Department, University Carlos III of Madrid, SPAIN (phone: +34 916249424; fax: +34 916249129; e-mail:jperalta@inf.uc3m.es)

Xiaodong Li. School of Computer Science and Information Technology, RMIT University. AUSTRALIA (phone: +61 3 99259585; fax: +61 3 96621617; e-mail: xiaodong.li@rmit.edu.au)

German Gutierrez. Computer Science Department, University Carlos III of Madrid, SPAIN (phone: +34 916249135; fax: +34 916249129; e-mail: ggutierr@inf.uc3m.es)

Araceli Sanchis. Computer Science Department, University Carlos III of Madrid, SPAIN (phone: +34 916249423; fax: +34 916249129; e-mail: araceli.sanchis@uc3m.es).

This contribution reports the methodology to carry out the automatic design of ANN that tackles the forecasting of a referenced set of time series [4]. The task will consist of forecasting several time series, not all of them with the same ANN, but an automatic method will be used to obtain a different ANN to forecast each time series.

Two different steps, as it was explained in an earlier work [5], will be done to get an ANN to forecast each time series. The first step will consist of setting the kind of ANN that will solve the forecasting task (Multilayer Perceptron), and the learning algorithm used (Backpropagation).

In the second step the design of the ANN will be done, setting the parameter values of the ANN, i.e. number of input nodes, number of hidden nodes, learning rate for BP and finally all connections weights. These parameters are established by carrying out a search process performed by two different evolutionary algorithms, a Genetic Algorithm (GA), and Differential Evolution algorithm (DE).

The paper is organized as follows. Sec II reviews the related work about how to tackle forecast task with ANN, and design of ANN with Evolutionary Computation. Sec III will explain how our system designs ANN with GA and DE to forecast time series. In Sec IV experimental setup and results are shown. And finally, conclusions and future works are described in Sec V.

II. Related work

A. Time series and ANN

Several works have tackled the forecasting time series task with ANN, not only computer science researchers, but statistics as well [1]. This shows the full consideration of ANN (as a data driven learning machine) into forecasting theory [6].

Before using an ANN to forecast, it has to be designed, i.e. establishing the suitable value for each freedom degree of the ANN [7] (kind of net, number of input nodes, number of outputs neurons, number of hidden layer, number of hidden neurons, the connections from one node to another , connection weights, etc). The design process is more an "art" based on test and error and the experience of human designer, than an algorithm. In [6] Zhang, Patuwo and Hu present a "state of the art" of ANN into forecasting task, in [8] is proposed an "extensive modeling approach" to review several designs of ANN.

The problem of forecasting time series with ANN is considered as modeling the relationship of the value of the element in time *"t"* (due to the net will only have one output

neuron) and the values of previous elements of the time series (*t-1, t-2,.., t-k*) to obtain a function as it is shown in (1):

$$a_t = f(a_{t-1}, a_{t-2}, \ldots, a_{t-k}) \quad (1)$$

B. ANN and Evolutionary Computation

Several works show methods to obtain ANN design by an automatic way; among them, those that use Evolutionary Computation (EC) reveal that the search process carried out by evolutionary techniques, obtain good results [9,10,11,12,13].

Some of them use Direct Encoding Schemata (DES) [9,10], others use Indirect Encoding Schemata (IES) [11,12,13]. For DES the chromosome contains information about parameters of the topology, architecture, learning parameters, etc. of the Artificial Neural Network. In IES the chromosome contains the necessary information so that a constructive method gives rise to an Artificial Neural Network topology (or architecture). Abraham [14] shows an automatic framework for optimization ANN in an adaptive way, and Xin Yao et. al. [15] try to spell out the future trends of the field. Just a couple of studies have been done using ANN and DE to generate a hybrid system to forecast time series [16,17].

III. ANN DESIGN WITH GA AND DE

A. Learning pattern set

In order to obtain a single ANN to forecast time series values, an initial step has to be done with the original values of the time series, i.e. normalize the data. And once the ANN gives the resulting values, the inverse process is carried out in order to evaluate the forecasting carried out by the ANN. This step is important as the ANN works with real values number between 0 and 1 as input values.

Furthermore, the time series known values will be transformed into a pattern set, depending on the k inputs nodes of each ANN generated in search process carry out by GA or DE. Therefore, each pattern consists in:

- "k" inputs values, that correspond to "k" normalized previous values of period t: $a_{t-1}, a_{t-2}, \ldots, a_{t-k}$.
- One output value, that corresponds to normalized time series value of period t.

This patterns set will be used to train and validate each ANN generated during the GA or DE execution. So patterns set will be split into two subsets, train and validation. The first x% from the total patterns set will generate the train patterns subset, and the validation subset will be obtained from the rest of the complete patterns set.

B. ANN design carried out with GA

The problem of designing ANN could be seen as a search problem into the space of all possible ANN. Moreover, that search can be done by a GA [18] using exploitation and exploration.

Therefore there are three crucial issues: i) the solution's space, what information of the net is previously set and what is included into the chromosome; ii) how each solution is codified into a chromosome, i.e. encoding schema; iii) and what is being looked for, translated into the fitness function.

In this approach it has been chosen Multilayer Perceptron (MLP) as computational model due to its approximation capability and inside this group, Full Connected MLP with only a hidden layer and Backpropagation (BP) as learning algorithm, according to [19]. This is because ANN with only one hidden layer are faster to be trained and easier to work than two or more hidden layer MLP.

As it was mentioned before the design of the ANN will be done by setting the parameter values of the ANN. In the case of MLP with only one hidden layer and BP, the parameters are: number of inputs nodes, number of hidden neurons, number of output neurons, (only one, it is set by the forecasting problem), which is the connection patterns (how the nodes are connected), and the whole set of connection weights (implemented by the seed used to initialize the connection weights as it will be explained later).

For our approach [5] to design ANN to forecast time series, a Direct Encoding Schema for Full Connected MLP has been considered. For this Direct Encoding Scheme the information placed into the chromosome will be: two decimal digits, i.e. two genes, to codify the number of inputs nodes (i); other two for the number of hidden nodes (h); two more for the learning factor (α); and the last ten genes for the initialization seed value of the connection weights (s) (seed in SNNS [20] is *"long int"* type, that is why it has been used 10 genes (decimal digits) to encode "s"). This way, the values of "i", "h", "α" and "s" are obtained from the chromosome as it can be seen in eq (2):

$$\begin{aligned} &\text{Chrom}: \\ &g_{i1}\ g_{i2}\ g_{h1}\ g_{h2}\ g_{\alpha 1}\ g_{\alpha 2}\ g_{s1}\ g_{s2}\ g_{s3}\ g_{s4}\ g_{s5}\ g_{s6}\ g_{s7}\ g_{s8}\ g_{s9}\ g_{s10} \\ &0 \le g_{xy} \le 9,\ x = i, h, \alpha,\ y = 1..10 \\ &i = \text{max\_inputs} \cdot ((g_{i1} \cdot 10 + g_{i2})/100) \\ &h = \text{max\_hiddens} \cdot ((g_{h1} \cdot 10 + g_{h2})/100) \\ &\alpha = ((g_{\alpha 1} \cdot 10 + g_{\alpha 2}))/100 \\ &s = g_{s1}\ g_{s2}\ g_{s3}\ g_{s4}\ g_{s5}\ g_{s6}\ g_{s7}\ g_{s8}\ g_{s9}\ g_{s10} \end{aligned} \quad (2)$$

The search process (GA) will consist of the following steps:

1. A randomly generated population, i.e a set of randomly generated chromosomes, is obtained.
2. The phenotypes (i.e. ANN architectures) and fitness value of each individual of the actual generation is obtained. To obtain the phenotype associated to a chromosome and its fitness value:

2.1 The phenotype) of an individual *"i"* of the actual generation is obtained.

2.2 Then, the train and validation patterns subsets for this neural net "*i*"are obtained from time series data , depending on the number of inputs nodes of the net "*i*", as it was commented above in section III.A.

2.3 The net is trained with BP (. The architecture (topology and connections weights set) of the net when the validation error (i.e. error for validation patterns subset) is minimum during the training process is saved (i.e. early stopping). So this architecture is the final phenotype of the individual.

3. Once that fitness value for whole population is already obtained, the GA operators as Elitism, Selection, Crossover and Mutation are applied in order to generate the population of the next generation, i.e. set of chromosomes.
4. The steps 2 (i.e. 2.1, 2.2, 2.3) and 3 are iteratively executed till a maximum number of generations are reached.

A schema of the whole search process can be seen at fig. 1.

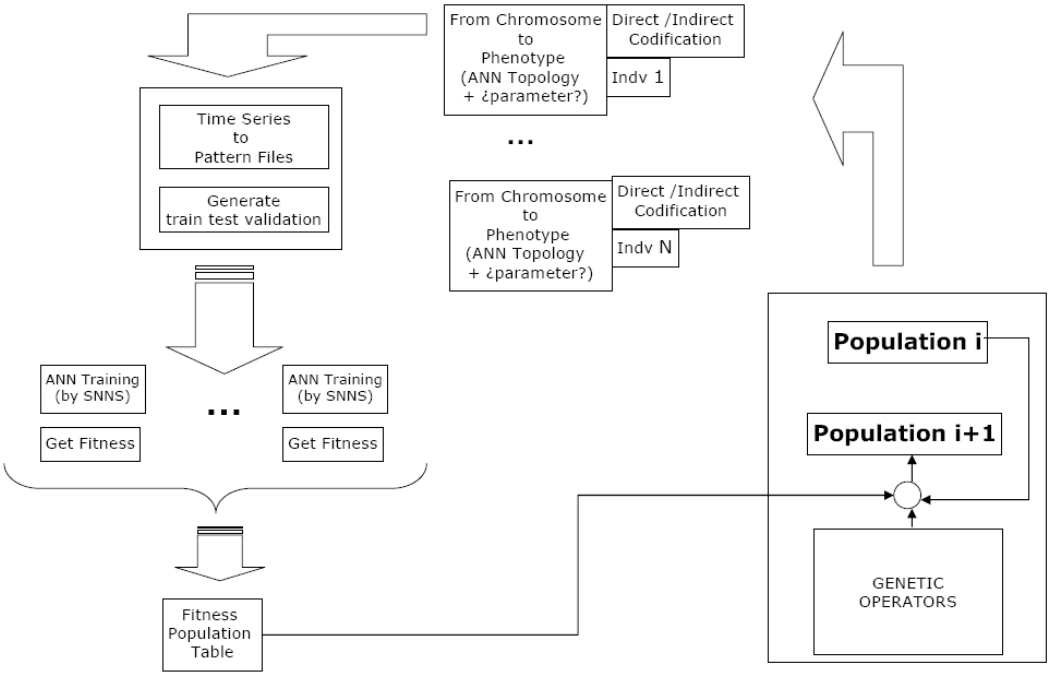

Fig. 1. ANN design by GA schema

The fitness value for each individual will be then the minimum validation error during the learning process (training of ANN topology), as it can be seen in eq (3):

$$fitness\ function = minimum\ validation\ error \quad (3)$$

The parameters for the GA are: population size, 50; maximum number of generations, 100; percentage of the best individuals that stay unchangeable to the next generation (percentage of elitism), 10%; crossover: parents are split in one point randomly selected, offspring are the mixed of each part from parents; mutation probability will be one divided between the length of the chromosome ($1/length\_chrom) = 1/16 \approx 0.7$), and it will be carried out for each gen of each chromosome.

Once that GA reaches the last generation, the best individual (i.e. ANN) from the last generation is used to forecast the future (and unknown) time series values.

The future unknown values (a_{t+1}) will be forecasted one by one using the k previous known values ($a_t, a_{t-1}, ..., a_{t-k}$). k is the number of inputs of the ANN obtained from the GA execution (i.e. the best ANN from the search process) To forecast several consecutive values ($a_{t+1}, a_{t+2,...}$) every time a new value is forecasted, it will be included in order into the previous known values set of the time series and used to forecast the next one.

C. *Differential Evolution algorithm (DE)*

Differential Evolution algorithms (DE), is a stochastic nonlinear optimization algorithm by Storn and Price [21]

and belongs to the class of evolution strategy optimizers. DE tries to find the global minimum of a multidimensional, multimodal (i.e. exhibiting more than one minimum) function with good probability. DE community has been growing since the mid 1990s and today more researchers are working on and with DE [16,22].

The crucial idea behind DE is a scheme for generating trial parameter vectors. DE differs from other evolutionary algorithms (EA) in their mechanism of generating offspring. In GA, an individual plays the role of a parent to generate an offspring. Nevertheless, DE adds the weighted difference between two vectors, i.e. chromosomes of the population, to a third vector. This way no separate probability distribution has to be used which makes the scheme completely self-organizing.

In DE a population of solution vectors is successively updated by addition, subtraction, and component swapping, until the population converges, hopefully to the optimum. No derivatives are used, very few parameters are set and it is a simple and apparently very reliable method. DE managed to finish 3rd at the First International Contest on Evolutionary Computation (1st ICEO). DE turned out to be the best genetic type of algorithm for solving the real-valued test function.

Just a couple of studies using DE have been used in weather time series forecasting field [16,17].
Due to this, a total automatic method, using advantages of DE and ANN (i.e. hybrid system), in which the user doesn't need to be an expert, to forecast all kind of time series, not only weather ones, is presented here.

D. ANN design carried out with DE

There are many schemes of generating individuals in DE. In this document, it is presented the simplest and most popular scheme, the DE/rand/1/bin, which is used in this research. Here we have the process for a DE/rand/1/bin:

1. $P_{x,g}$ <= Generate and evaluate an initial population of x solutions where g is the number of dimensions in the search-space.
2. Repeat for i = 0, 1, 2, …, until a stopping criterion is met
    a. A target vector (X_i) and a base vector (X_{r0}) are chosen.
    b. Two random different population members (X_{r1}, X_{r2}) are also chosen.
    c. Compute weighted difference vector from X_{r1} and X_{r2}.
    d. Multiply "c" by the mutation factor *F*.
    e. V_i <= Add "d" to base vector (X_{r0}) to obtain the mutant population $P_{v,g}$.
    f. U_i <= Crossover between target vector (X_i) and V_i.
    g. Selection is carried out between X_i and U_i.

As it can be seen at the previous pseudocode, different steps have to be done during DE process. First we start with a randomly chosen solution vectors as an initial population of NP possible solutions ($P_{x,g}$). As it was commented above a target vector (X_i), a base vector (X_{r0}), and two random different population members (X_{r1}, X_{r2}) are also chosen. For each X_i in (1,… ,NP), it is formed a "*mutant vector*" (V_i) using the other three elements (X_{r0}, $X_{r1}$and X_{r2}) as it is shown in (4):

$$V_i = X_{r0} + F(X_{r1} - X_{r2}) \tag{4}$$

Where r0, r1, and r2 are three mutually distinct randomly drawn indices from (1, …,NP), and also distinct from i, and mutation factor values is 0<F<=2. Fig. 2 shows the process of forming the *"mutant vector"*.

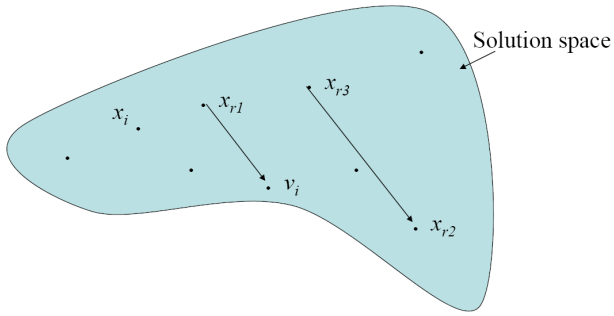

Fig. 2. Forming the mutant vector

To crossover X_i and V_i to form the trial vector U_i, for each component of vector, it is drawn a random number in [0,1] called $rand_j$. Let the crossover ratio (i.e. CR) be a cutoff with value 0<=CR<1. If $rand_j$<=CR, Ui_j= Vi_j, else U_{ij}= X_{ij}. To ensure at least some crossover, one component of U_i is selected at random to be directly from V_i. For example, if we have X_i and V_i as follow:

$$X_i = (X_{i1}, X_{i2}, X_{i3}, X_{i4}, X_{i5})$$
$$V_i = (V_{i1}, V_{i2}, V_{i3}, V_{i4}, V_{i5})$$

U_i could be:

$$U_i = (V_{i1}, X_{i2}, X_{i3}, X_{i4}, V_{i5})$$

Where index 1 (V_{i1}) of U_i has been randomly selected as definitive crossover from V_i. V_{i5} has been chosen from V_i because $rand_5$ was lower than CR, and in the rest of the cases, $rand_i$ was greater than CR. Fig. 3 shows crossover process.

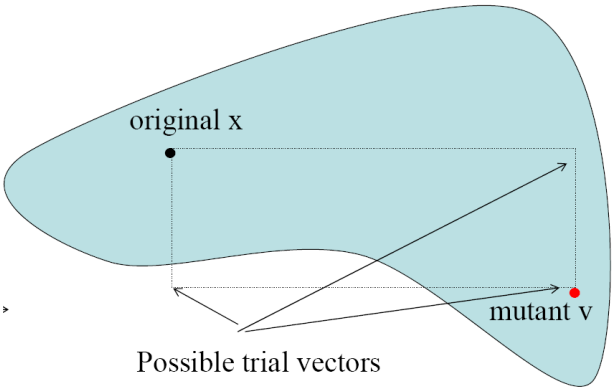

Fig 3. Crossover X_i and V_i to form the trial vector

Once crossover is finished, selection has to be done. The idea of selection in this algorithm is simple, if the new offspring (U_i) is better (i.e. better fitness value) than the target vector (X_i) then, U_i pass to the next generation, otherwise it is the target vector who will be again in the next generation. Fig. 4 shows DE whole process.

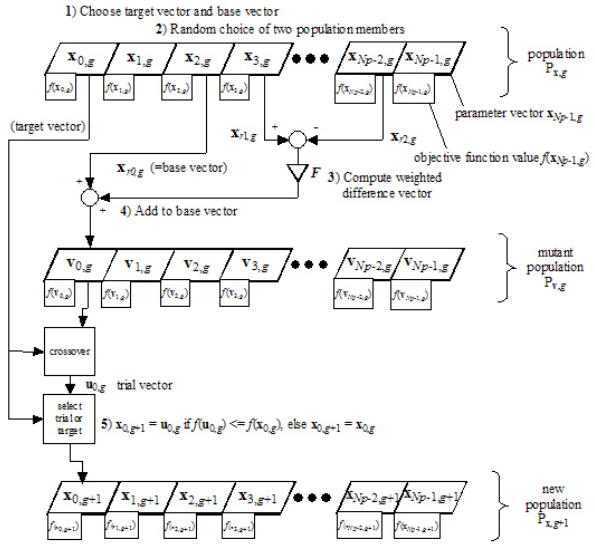

Fig. 4. DE schema

Equation 5 shows a summary of DE algorithm:

$$U_i=U_{j,i}=\begin{cases} X_{j,r0} + F(X_{j,r1} - X_{j,r2}) \text{ if } (rand_j < CR \text{ or } j = j_{rand}) \\ X_{j,i} \text{ otherwise} \end{cases} \quad (5)$$

If we also have a look to the process for a general GA, it would be:

1. D_0 <= Generate and evaluate an initial population of solutions
2. Repeat for k = 0, 1, 2, …, until a stopping criterion is met
   a. $D_k^{Elitism}$ <= Select a subset of solutions from D_k
   b. $D_k^{Crossover}$ <= Apply crossover to solutions from D_k
   c. $D_k^{Mutation}$ <= Apply mutation to solutions from $D_k^{Crossover}$
   d. D_{k+1} <= Create the new population with solutions from $D_k^{Mutation}$ and $D_k^{Elitism}$
   e. Evaluate solutions in D_{k+1}

To apply DE to our system it was necessary to replace the GA, who is responsible of carrying out the global search into the hybrid system, for DE.

It can be observed that the principal differences between GA and DE consist of steps *"a", "b", "c", "d"* and *"e"*, and it was also necessary to add two new steps *"f"* and *"g"* where the new selection process is carried out.

IV. Experimental Setup and Results

A. Time Series

Five time series will be used to evaluate our methods. These time series are named Passengers, Temperature, Dow-Jones, Quebec, and Mackey-Glass [23]. Passengers time series has the information about the number of passengers of an international airline in thousands, measured monthly from January of 1949 till December of 1960, the source is Box & Jenkins (1976). Temperature time series shows the mean monthly of air temperature measured at Nottingham Castle from 1920 till 1939; in this case the source is O.D. Anderson (1976). Dow-Jones is about the monthly closings of the Dow-Jones industrial index from August of 1968 till August of 1981, the source is Hipel and Mcleod (1994). Quebec represents the number of births daily measured in Quebec from 1st of January of 1977 till 31 of December of 1978. And the last one called Mackey-Glass is based on the Mackey-Glass differential equation and is widely regarded as a benchmark for comparing the generalization ability of different methods. This series is a chaotic time series generated from a time-delay ordinary differential equation.

B. Experimental setup

The time series values have to be rescale, into the numerical range value [0,1], considering not only the known values, but the future values (those to be forecasted).

So, the maximum and minimum limits for normalizing (*max4norm, min4norm* respectively) cannot be just the maximum (*max*) and minimum (*min*) known time series values. A margin from max and min has to be set if future

values were higher or lower than known values already are. This margin will depend on another parameter (*Prct_inc*). In those cases in which the time series is stationary a *Prct_inc* of 10% will be enough, but when the time series is increasing or decreasing *Prct_inc* should be at least of 50%. As it could be forecasted new values for a time series that will rise of a fall, it is needed a enough big margin so the new values, obtained as output of ANN, can be into the numerical range [0,1]. This Equation (5) shows how are obtained max4norm and min4norm.

$$\begin{aligned} \text{max4norm} &= \text{max} + (\text{Prct\_inc} \cdot (\text{max} - \text{min})) \\ \text{min4norm} &= \text{min} - (\text{Prct\_inc} \cdot (\text{max} - \text{min})) \end{aligned} \quad (5)$$

C. GA versus DE

Both ways to forecast time series, i.e both hybrid systems, one with GA and other with DE, have been executed five times (200 generations each time) for each time series. For each time series, the average result (and standard desviation) of the five simulations is shown.

To evaluate the error for each method, forecasted values (i.e. test set, not train or validation sets) are compared with real values. Two sort of errors on forecasted values are used: MSE (mean squared error) and SMAPE (symmetric mean absolute percent error [4]);SMAPE has been used at NN3 and NN5 forecasting competitions. Results are shown in Tables I and II.

In Table I, it is shown the results obtained for Passengers, Temperature, Dow-Jones, Quebec and Mackey-Glass time series in generation number 100. In this table, the columns show: MSE and SMAPE error in forecasting (i.e. test set) for each time series. These errors are relative to the average of the five times experiments have been run, choosing each execution the best individual from the last generation of the GA or the DE.

In Table II, it is shown the results obtained for Passengers, Temperature, Dow-Jones, Quebec and Mackey-Glass time series in generation number 200. In this table, the columns will show: MSE and SMAPE error in forecasting (i.e. test set) for each time series. These errors are relative to the average of the five times experiments have been run, choosing each execution the best individual from the last generation of the GA or the DE.

TABLE I
SMAPE AND MSE PASSENGERS, TEMPERATURE, DOW-JONES, QUEBEC AND MACKEY-GLASS WITH GA AND DE FOR 100 GENERATIONS

100 Generations		GA	DE
Passengers	SMAPE (%)	3.180	3.358
	MSE	0.00061	0.00065
Temperature	SMAPE (%)	4.308	3.907
	MSE	0.00358	0.00294
Dow-Jones	SMAPE (%)	6.662	8.188
	MSE	0.02150	0.03229
Quebec	SMAPE (%)	12.643	13.739
	MSE	0.02540	0.02662
Mackey-Glass	SMAPE (%)	8.672	5.988
	MSE	0.00363	0.00175

TABLE II
SMAPE AND MSE PASSENGERS, TEMPERATURE, DOW-JONES, QUEBEC AND MACKEY-GLASS WITH GA AND DE FOR 200 GENERATIONS

200 Generations		GA	DE
Passengers	SMAPE (%)	3.148	3.118
	MSE	0.00058	0.00058
Temperature	SMAPE (%)	4.239	3.907
	MSE	0.00347	0.00294
Dow-Jones	SMAPE (%)	6.307	5.810
	MSE	0.01993	0.01735
Quebec	SMAPE (%)	12.121	13.682
	MSE	0.02149	0.02663
Mackey-Glass	SMAPE (%)	8.042	3.744
	MSE	0.00309	0.00064

As it can be observed in Table I, applying DE instead of GA to these time series doesn't achieve better forecasting (MSE/SMAPE) in many of the time series when the experiment has been run only 100 generations. Just Mackey-Glass and Temperature obtain a better SMAPE result with DE and in Mackey-Glass case; the improvement is about 2.6%.

But if the experiment is run over 200 generations, it can be seen in Table II an important improvement in almost all the time series, where DE obtain a better forecast than GA in four of the five time series. Only in Quebec time series GA is still better than EDA although both results are close. A special consideration has to be taken on Mackey-Glass time series where the SMAPE error result is 3.744%, being the values forecasted by our approach almost identical to the real time series values.

The better results obtained by DE, compared with GA, after having run the experiments 200 generations could be explained because in DE, more variation in population (because solution has not converged yet) leads to more varied search over solution space. That is why it can take more time to DE to arrive to a better solution.

To have a better idea about the forecast of each time series and how close to the real values each forecasting

method was, a graph for each time series showing all the forecasts done by each method will be carried out. Figure 5 shows Passengers forecast for each method. Figure 6 shows Temperature, Figure 7 shows Dow-Jones, Figure 8 shows Quebec and Figure 9 Mackey-Glass. A zoom of Quebec will be done in Figure 10. A zoom of Mackey-Glass will be done in Figure 11. TS(tr+val) represents the known values of the time series with which we have worked and TS(test) shows the future unknown real values that have to be forecasted (i.e. test subset).

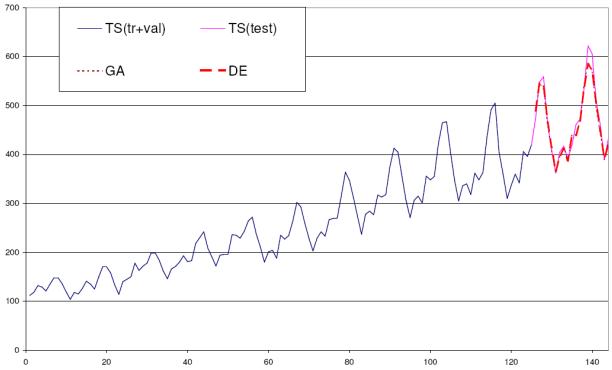

Fig. 5. Passengers forecast with GA and DE

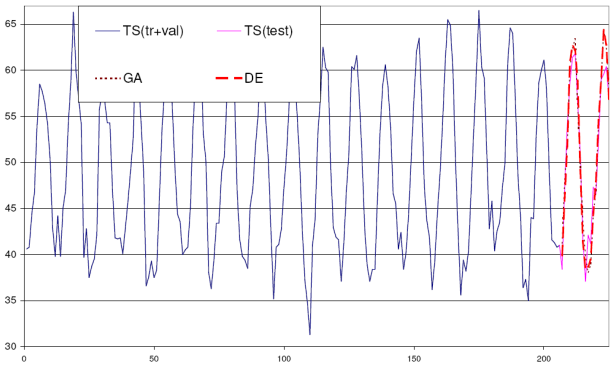

Fig. 6. Temperature forecast with GA and DE

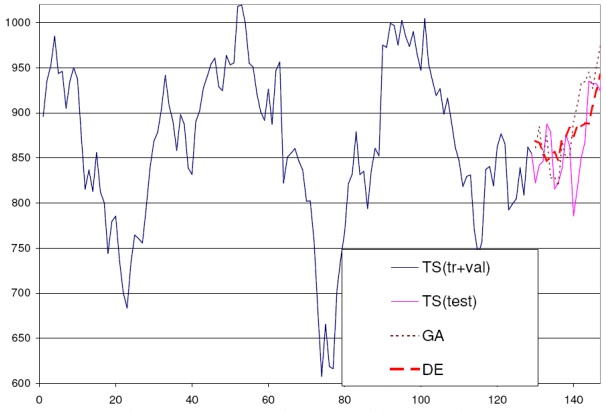

Fig. 7. Dow-Jones forecast with GA and DE

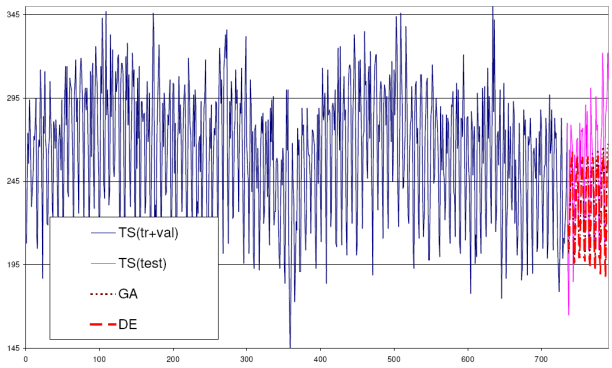

Fig. 8. Quebec forecast with GA and DE

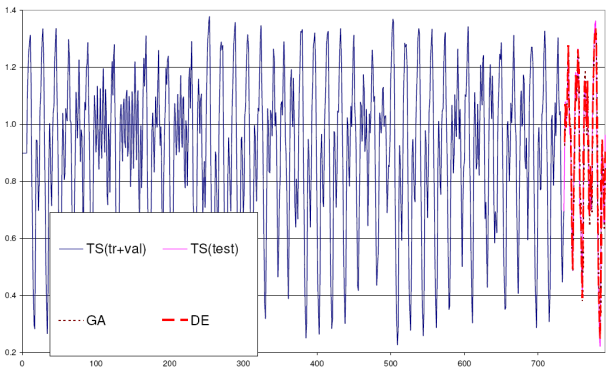

Fig. 9. Mackey-Glass forecast with GA and DE

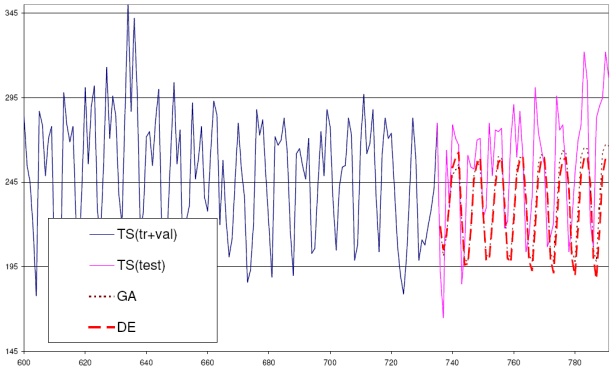

Fig. 10. Zoom of Quebec forecast with GA and DE

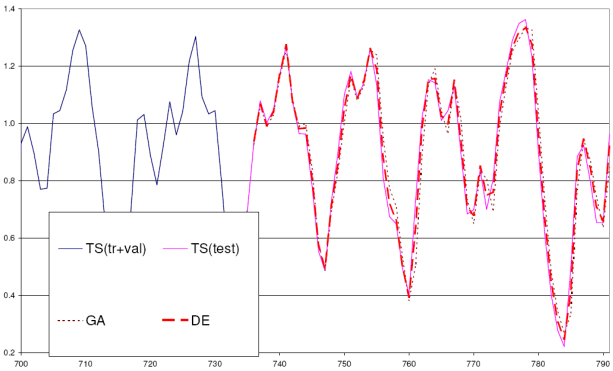

Fig. 11. Zoom of Mackey-Glass forecast with GA and DE

V. CONCLUSIONS AND FUTURE WORKS

The results of the experiments disclose that using DE instead of GA obtain different results, depending on the number of generations they are executed. With only 100 generations, DE results don't improve too much compared to GA. But if 200 generations are reached, it can be observed a significant improvement, some times with a gain of 4.3% in the results, as it happens in Mackey-Glass time series.

As it was commented before, obtaining better results by DE than with GA after having run the experiments 200 generations could be explained because in DE more variation in population (because solution has not converged yet) leads to more varied search over solution space. That is why it can take more time to DE to arrive to a better solution.

As it is a totally automatic method, it will not be necessary any previous knowledge from the user. So the user will not have to be an expert in time series, statistics, mathematics or computational intelligence. The user just have to give the time series he wants to forecast and the number of future elements he wants to be forecasted to the system; and this method will give these forecasted values as result to the user.

This approach was presented as an automatic method to design ANN in NN5 competition, getting the 6th position with SMAPE error of 21.9% in Neural Nets and Computational Intelligence methods (NNCI) ranking, for the reduced dataset (i.e. 11 time series). Best result on NNCI ranking and reduced data was a SMAPE error of 19.0%. Autobox tool [24] based on Box-Jenkins forecasting methodology got an error of 23.9%.

Future works with additional time series, with similar characteristics to Quebec, Mackey-Glass will allow us to obtain more accurate conclusions about the effect of using DE in stead of GA. On the other hand, it would be really interesting to try to improve the system with some ideas like: in stead of using a random X_{r0}, use the best one (i.e. the one with the best fitness value); or instead of using single difference (i.e. X_{r1}-X_{r2}), to use more vectors for more variation, for example (X_{r1}-X_{r2}+X_{r3}-X_{r4}).

Other interesting future works are: to use *"cross validation"* into the GA for a better evaluation of each individual; using sparsely connected ANN to try to improve the forecast to obtain an accurate system.

REFERENCES

[1] Spyros G. Makridakis, Steven C. Wheelwright, Rob J Hyndman. Forecasting: Methods and Applications.

[2] Ian Nunn, Tony White."The application of antigenic search techniques to time series forecasting". Genetic and Evolutionary Computation Conference 2005. ISBN:1-59593-010-8.

[3] Paulo Cortez, José Machado, José Neves. "An evolutionary artificial neural network time series forecasting system". IASTED 1996.

[4] Time Series Forecastig Competition for Neural Networks and Computational Intelligence. http://www.neural-forecasting-competition.com. Accessed on October 2008.

[5] J. Peralta, G. Gutierrez, A. Sanchis, "ADANN: Automatic Design of Artificial Neural Networks". ARC-FEC 2008 (GECCO 2008). ISBN 978-1-60558-131-6.

[6] Zhang, G.; Patuwo, B.E. & Hu, M.Y. Forecasting with artificial neural networks: The state of the art International Journal of Forecasting, 1998, 14, 35-62.

[7] Haykin, S. Simon & Schuster (ed.) Neural Networks. A Comprehensive Foundation Prentice Hall, 1999.

[8] Crone, S. F. Stepwise Selection of Artificial Neural Networks Models for Time Series Prediction Journal of Intelligent Systems, Department of Management Science Lancaster University Management School Lancaster, United Kingdom, 2005.

[9] T. Ash. Dynamic Node Creation in Backpropagation Networks ICS Report 8901, The Institute for Cognitive Science, University of California, San Diego (Saiensu-sh, 1988), 1988.

[10] D.B. Fogel, Fogel L.J. and Porto V.W. Evolving Neural Network, Biological Cybernetics, 63, 487-493, 1990.

[11] Gruau F. "Genetic Synthesis of Boolean Neural Networks with a Cell Rewriting Developmental Process". Proceedings of COGANN-92 International Workshop on Combinations of Genetic Algorithms and Neural Networks, pp. 55-74, IEEE Computer Society Press, 1992.

[12] Yao, X. and Lin, Y. A new evolutionary system for evolving artificial neural networks, Transactions on Neural Networks, 8(3): 694-713, 1997.

[13] Kitano, H.: Designing Neural Networks using Genetic Algorithms with Graph Generation System, Complex Systems, 4, 461-476, 1990.

[14] Ajith Abraham, Meta-Learning Evolutionary Artificial Neural Networks, Neurocomputing Journal, Elsevier Science, Netherlands, Vol. 56c, pp. 1-38, 2004.

[15] X. Yao (1993), "A review of evolutionary artificial neural networks", International Journal of Intelligent Systems, 8(4):539-567.

[16] R. A. Araujo, G. C. Vasconselos and A. E. Ferreria. Hybrid differential evolutionary system for financial time series forecasting. 2007.

[17] H. M. Abdul-Kader. Neural networks training based on differential evolution algorithm compared with other architectures for weather forecasting34. 2009.

[18] Fogel, D. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. Wiley-IEEE Press, 1998.

[19] G. Cybenko. Approximation by superposition of a sigmoidal function. Mathematics of Control, Signals and Systems, 2, 303-314, 1989.

[20] Prof. Dr. Andreas Zell,, WSI Computer Science Department, Computer Arquitecture, Software, Artificial Neural Networks http://www-ra.informatik.uni-tuebingen.de/SNNS/

[21] Storn, Rainer, and Kenneth Price. Differential Evolution –A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization11, 1997, pp. 341-359.

[22] Wickramasinghe, W. and Li, X. (2008), "Choosing Leaders for Multi-objective PSO Algorithms using Differential Evolution", in Proceedings of the seventh International Conference on Simulated Evolution and Learning (SEAL'08), Lecture Notes in Computer Science (LNCS 5361), Springer, p.249 - 258.

[23] Hyndman, R.J. (n.d.) Time Series Data Library, http://www.robjhyndman.com/TSDL. Accessed on February 1st 2010.

[24] Automatic Forecasting Systems. http://www.autobox.com. Accessed on October 2008.